7П

УДК 004.891.3

А.С. Пригожев
Одесский национальный политехнический университет, г. Одесса, Украина
prigozhev@rambler.ru

Визуализация данных для анализа программного обеспечения с использованием экспертной системы

В статье предлагается формальная модель для автоматизированного анализа программного обеспечения в процессе отладки и тестирования при помощи экспертной системы. Рассматриваются вопросы интеграции такой модели в экспертную систему поддержки разработчика. Предлагаемая модель позволяет в дальнейшем разработать формальные правила, позволяющие выявлять наиболее существенные ошибки в функционировании программного обеспечения.

Введение

Отладка современного программного обеспечения (ПО) является весьма трудоёмкой задачей в силу довольно большого объёма кода такого программного обеспечения. Также довольно часто ПО разрабатывается на разных, пусть и достаточно близких языках программирования. В таких случаях процесс отладки существенно затруднён, т.к. значительная часть работы выполняется программистами вручную. Поэтому необходимо автоматизировать процесс отладки с целью упрощения поиска ошибок в программном обеспечении.

Существующие средства автоматизации отладки и тестирования программного обеспечения позволяют выполнять в автоматизированном режиме основные операции, связанные с данными процессами. Большинство современных средств разработки позволяет визуализировать значения переменных в процессе отладки на каждом шаге алгоритма. Недостатком данного подхода является слабая реализация возврата к предыдущим шагам программы. Необходимость такого возврата возникает в процессе отладки достаточно часто, поскольку для принятия решения о наличии ошибки в программе приходится знать предыдущие значения переменных.

Существующие средства автоматизации тестирования, основанные на графовых и автоматных моделях, позволяют решать рассматриваемые задачи с учётом дополнительной информации, представляемой разработчиком. Поскольку такая информация часто зависит от понимания самим разработчиком задачи, то процесс синтеза тестов может быть неточным.

Поэтому существует задача дальнейшего развития средств отладки и тестирования. Одним из направлений развития этих средств может являться применение экспертных систем, основанных на представлении знаний о задачах в виде графа, для отладки и тестирования программного обеспечения. Использование таких средств позволяет отлаживать и тестировать программу не только пошагово, но и при помощи экспертных правил.

Однако для решения задачи тестирования и отладки недостаточно только знаний о порядке решения задач и действий по их решению. Необходимо разработать также формальную модель, позволяющую представлять значения данных на каждом этапе исполнения программы и анализировать их, а также разработать структуру экспертной системы для анализа данных. Решение этих задач является целью данной статьи.

Модель для визуализации данных

Для реализации процесса анализа исполнения исходного кода предлагается использовать автоматизированный анализ значений переменных в ходе исполнения программы, привязанный к определённому моменту времени. Использование такого подхода позволит в дальнейшем формализовать правила для анализа программного кода в системе.

Состоянием программы в определённый момент времени будем называть совокупность всех доступных в данный момент переменных, их значений и некоторого дополнительного набора параметров. Если номер шага программы обозначить буквой t, то соответствующее состояние программы будем обозначать $S(t)$. Таким образом, состояние программы является функцией от времени. Состояние программы включает в себя множество состояний всех доступных в данный момент переменных программы. Видимость переменных в программе зависит от местоположения в программе текущего оператора.

Все переменные с точки зрения проводимого анализа разделяются на три класса: переменные типа «данные», переменные типа «указатели» и переменные смешанного типа [1]. Кроме этого, необходимо ввести дополнительные переменные, принадлежащие соответствующим классам, – аппаратные переменные, которые характеризуют взаимодействие с устройствами системы. Их тип и максимальное, и минимальное возможные значения зависят от типа устройства, для которого создаётся переменная.

Для каждого класса переменных системы вводится ряд дополнительных характеристик. В частности, рассматривается минимально и максимально возможные теоретические значения переменных в программе, возможные минимальные и максимальные проектные значения и размер переменных в программе. Обозначая теоретические минимальные и максимальные значения переменных $tx_{\min}$ и $tx_{\max}$, соответствующие им проектные значения $x_{\min}$ и $x_{\max}$, размер переменной и имя переменной как r и i соответственно, а значение как z, состояние переменной программы можно представить в виде следующей семёрки:

$$(i, tx_{\min}, tx_{\max}, x_{\min}, x_{\max}, r, z)\,. \qquad (1)$$

Множество всех шестёрок (1) для переменных класса «данные» будем обозначать буквой *D*, для переменных класса «указатели» – буквой *P*, для переменных смешанного класса – *Dp*.

Соответственно состояние переменных в некоторый момент времени может быть описано следующей двойкой:

$$S(t) = (t, (D, P, Dp))\,. \qquad (2)$$

Первая компонента двойки (2) является временной составляющей, в нашем случае – это номера шагов в программе. Вторая компонента – это тройка, описывающая состояния переменных на данном шаге. Множество таких компонент, упорядоченное по первой компоненте, образует вектор, аналогичный временному ряду [3].

Для проведения корректного анализа программного кода вершинам модифицированного графа задач должны быть сопоставлены элементы двойки (2). Прохождение дерева в прямом порядке, как указано в [1], эквивалентно выполнению программного кода, на основе которого было построено данное дерево. При каждом проходе вершины графа ей сопоставляется порядковый номер выполняемого действия алгоритма. Это позволяет организовать анализ, используя выполняемые операторы и обрабатываемые данные.

Каждую функцию в языке программирования можно представить в виде некоторой схемы в алгебре Дейкстры [4]. Таким образом, каждая функция представляется некоторым подграфом на графе задач. Число, соответствующее вершине вызова функции, является входом функции. С учётом свойств обхода графа в таблице функций можно выявить ряд распространённых ошибок, таких как использование нулевых указателей, деление на ноль и т.п. Например, если в функции существует локальный указатель, и по этому адресу записывается значение на шаге с номером меньшим, чем инициализация указателя, то можно сделать вывод об использовании неинициализированного указателя.

Структура экспертной системы поддержки разработчика

Построенная модель и дерево задач используются в экспертной системе поддержки разработчика «ExDev», структура которой приведена на рис. 1.

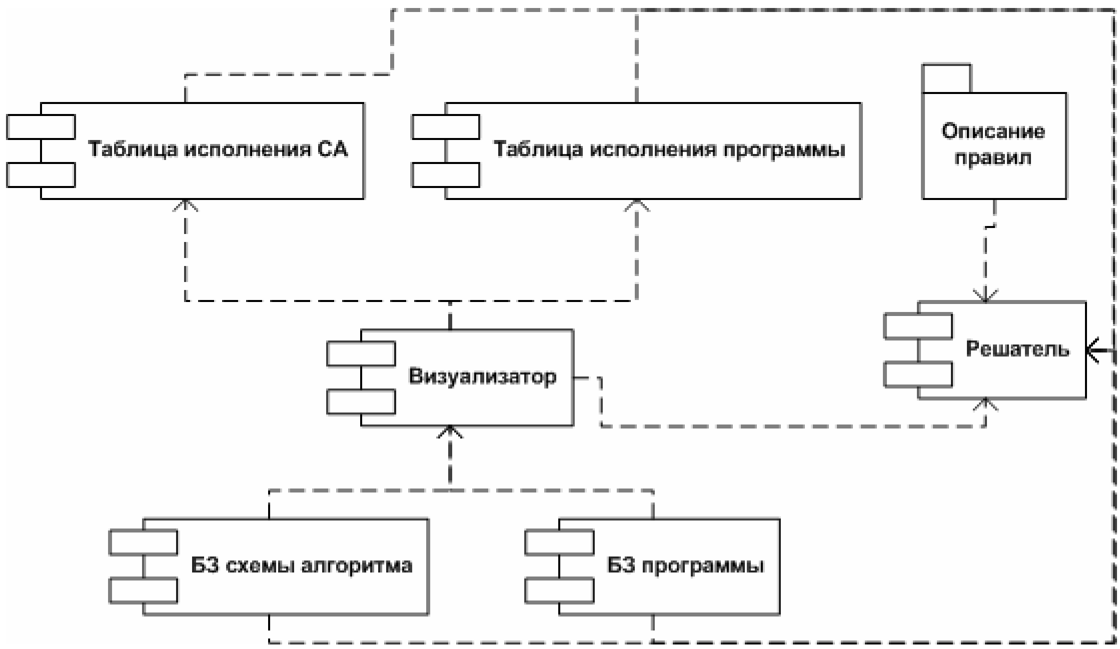


Рисунок 1 – Структура системы «ExDev» в нотации UML

Основой системы является решатель, который использует информацию, хранящуюся в базе знаний о программном коде и схеме алгоритма. В таблицах исполнения хранится информация, представленная в моделях (1) и (2).

Описание правил для решателя системы проводится в терминах моделей (1) и (2), а также сопоставленного с ним дерева задач. Компонент визуализатор связан с решателем и управляется им. Данный компонент позволяет построить модели (1) и (2) для конкретной программы либо схемы алгоритма. Использование компонента

совместно с решателем позволяет проводить предварительный анализ построенного дерева на предмет наличия семантических ошибок, таких, например, как бесконечное зацикливание. Вся информация как о дереве, так и о визуализации программы хранится в базе знаний экспертной системы.

Структура базы знаний для визуализации переменных в программе

Для выявления ошибок, допущенных в программе, необходимо разработать структуру базы данных для экспертной системы поддержки разработчика. В процессе визуализации алгоритма и выполняемой программы строится таблица визуализации, которая содержит большое количество повторяющейся информации. Также данная таблица должна быть связана непосредственно с деревом, которое хранит информацию о коде программы и алгоритма разработанной системы.

Таблица визуализации в памяти представляет собой структуру, содержащую либо информацию о текущем значении переменной, если эта переменная изменилась на данном шаге, либо ссылку на строку с текущим значением. Также данная структура содержит ссылку на соответствующий шаг алгоритма.

Для хранения базы знаний, включающей в себя дерево задач и таблицу визуализации, предлагается следующая структура реляционных таблиц (рис. 2).

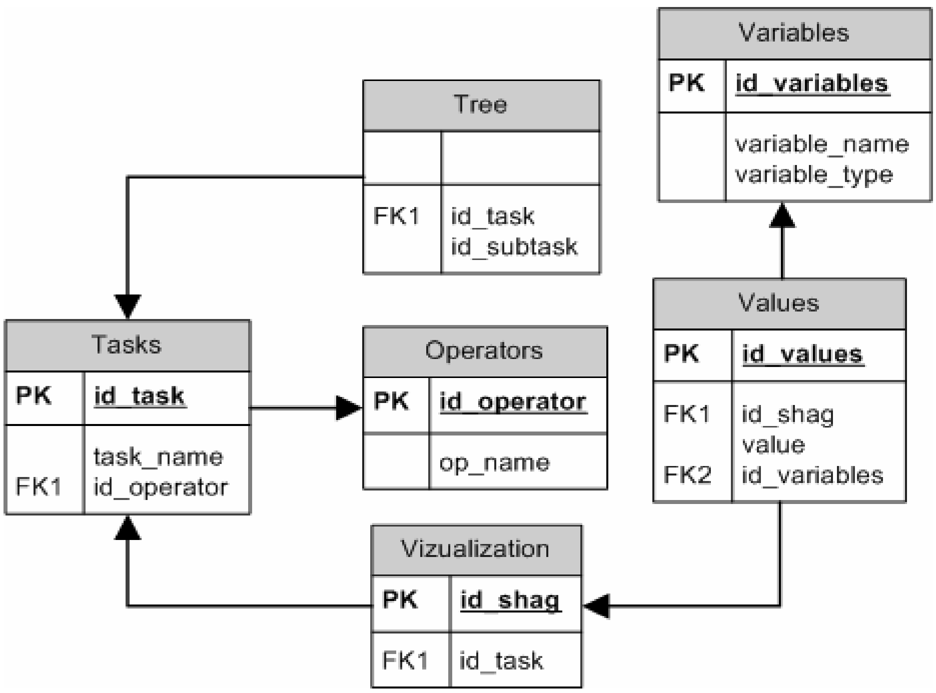


Рисунок 2 – Схема базы знаний для экспертной системы поддержки разработчика

В таблице «Tree» хранится непосредственно дерево, которое синтезировано для кода программы или алгоритма. Дерево в таблице «Tree» представляет собой множество записей, каждая из которых соответствует единичной ячейке матрицы смежности. Таблица «Tasks» содержит перечень задач, входящих в дерево. Если задача имеет подзадачи, то в поле id_operator записывается 0, в противном случае там содержится ссылка на записи в таблице операторов «Operators». По идентификатору задачи в

таблице визуализации «Vizualization» определяются номера шагов алгоритма, на которых решалась соответствующая задача. Таблицы «Variables» и «Values» хранят информацию о переменных, содержащуюся в моделях (1) и (2). Таблица «Variables» хранит сведения о переменных, которые не зависят от алгоритма, а таблица «Values» содержит сведения, зависящие непосредственно от алгоритма.

Выводы

В ходе исследования была решена задача построения модели для пошагового представления значений переменных в экспертной системе поддержки разработчика. Данная модель позволяет представить для анализа процесс выполнения алгоритма с учётом временной составляющей исполнения алгоритма. Ранее синтезированное представление алгоритма в виде дерева для экспертной системы поддержки пользователя [5] применено для задач тестирования и отладки программного обеспечения. Использование перечисленных моделей позволяет автоматизировать процессы тестирования и отладки с использованием экспертных систем.

Литература

1. Пригожев А.С. Языконезависимая среда разработчика для тестирования программного обеспечения / А.С. Пригожев // Радіоелектронні та комп'ютерні системи. – 2009. – № 7. – С. 225-230.
2. Пригожев А.С. Принципы построения интеллектуальной среды разработчика программного обеспечения на основе экспертной системы / А.С. Пригожев // Искусственный интеллект. Интеллектуальные системы: материалы IX Международной научно-технической конференции (пос. Кацивели, Крым, Украина, 22 – 27 сент. 2008). – Донецк : ИПИИ «Наука і освіта», 2008. – Т. 2. – С. 127-131.
3. Бриллинджер Д. Временные ряды: обработка данных и теория / Бриллинджер Д. – М. : Мир, 1980. – 532 с.
4. Цейтлин Г.Е. Введение в алгоритмику / Цейтлин Г.Е. – К. : Сфера, 1998. – 310 с.
5. Пригожев А.С. Информационная технология помощи пользователю / Пригожев А.С. // Холодильная техника и технология. – 2007. – № 2. – С. 98-101.

О.С. Пригожев

Візуалізація даних для аналізу програмного забезпечення з використанням експертної системи

У статті запропонована формальна модель для автоматизованого аналізу програмного забезпечення у процесі налагодження та тестування за допомогою експертної системи. Розглядаються питання інтеграції такої моделі в експертну систему підтримки розробника. Запропонована модель дозволяє у подальшому розробити формальні правила, які дозволяють виявляти найбільш суттєві помилки у функціонуванні програмного забезпечення.

A.S. Prigozhev

Data Visualization for the Software Analysis with Usage of a Consulting Program

In the paper the formal model for the software automated analysis during a debugging and testing by means of a consulting program is offered. Problems of model integration in a development engineer support expert system are considered. The offered model allows to develop in the further the formal rules, allowing to reveal the most essential errors in performance of the software.

Статья поступила в редакцию 30.06.2009.